Benchmarking communication middleware for cloud computing virtualizers

Marisol García-Valls, Pablo Basanta-Val, Rosbel Serrano-Torres
Distributed Real-Time Systems Laboratory
Departamento de Ingeniera Telemática
Universidad Carlos III de Madrid
Av. de la universidad 30
28911 Leganés, Madrid, Spain
{mvalls,pbasanta}@it.uc3m.es

***Abstract*—Virtualization technologies typically introduce additional overhead that is specially challenging for specific domains such as real-time systems. One of the sources of overhead are the additional software layers that provide parallel execution environments which reduce the effective performance given by the infrastructure. This work identifies the factors to be analysed by a benchmark for performance evaluation of a virtualized middleware. It provides the set of benchmark tests that evaluate empirically the overhead and stability on a trendy communication middleware, DDS (Data Distribution System for Real-Time), which enables message transmissions via publisher-subscriber (P/S) interactions. Two different implementations, RTI and OpenSplice, have been analysed over a general purpose virtual machine monitor to evaluate their behavior on a client-server application. Obtained results have provided initial execution clues on the performance that a virtualized communication middleware like DDS can exhibit.**

I. Introduction

Communications middleware and virtualization technologies have been two main contributions to the development and maintainability of software systems. On the one hand, middleware brings in the capacity to abstract the low level details of the networking protocols and the associated specifics of the physical platforms (e.g. endianness, frame structure, and packaging, among others). This augments the productivity of systems development by easing the programmability and debugging. More recently, virtualization technologies have promoted a new technological trend that has fast penetrated different domains due to the benefits that it brings about: a) speed up of the customized system development and deployment to specific platforms; b) server consolidation and the subsequent savings on energy, etc. ; c) reducing maintenance and deployment costs and d) data availability any time and anywhere.

Communication middleware and virtualization technology originated for general purpose distributed applications, so initially in a different perspective from that of real-time environments where determinism is a key target. As science evolves and new applications are envisioned and engineered, real-time applications have progressively approached middleware and virtualization technologies, facing the problem of temporal predictability. The traditional focus of real-time and middleware has been significantly different. Networked real-time systems traditionally have focused on eliminating (or minimizing) the sources of unpredictability by direct programming of tasks in the real-time operating system or directly in the hardware platform itself and using controlled medium access protocols to develop real-time networks. Middleware has typically been implemented for distributed systems over non collision-free networks, and using software engineering techniques that introduce additional software layers aiming at easing programmability and interoperability. As a consequence, communications middleware has appeared as a black box, containing extra code that is difficult to analyse with sufficient level of detail and guarantees as required by some real-time applications.

Over the past decade, the OMG's DDS standard [1] (Data Distribution Service for Real-Time Systems) has appeared with considerable success for distributed soft real-time applications. DDS provides an asynchronous interoperability via a publish-subscribe paradigm that is data-centric. One of the success factors of DDS is that it provides quality of service (QoS) communication by means of specifying a collection of diverse QoS parameters. There are different realizations of the DDS standard that achieve different behaviors, mainly with respect to performance and to the specific set of implemented QoS parameters. In general, the level of temporal guarantees provided by different implementations varies depending on different factors such as the physical deployment, application type, and middleware communication paradigm and fine-tuning. There are not many public independent studies about the performance achieved by the different implementations.

The performance of middleware can be essential for determining if a specific real-time application can be migrated to the cloud. This requires to analyse the timely behavior of the middleware implementation and extract conclusions about the suitability for specific physical deployments (i.e., software, hardware, and network structure) and application types (e.g. data intensive, sporadic short messages, etc.). Also, traditional virtualization techniques can be a source of overhead and even nondeterminism. Virtualization technology comes at the cost of, in general, being more prone to suffering variations in performance compared to bare machine execution, in general. The

latter needs to be studied for the specific deployments since technological developments, such as multicore systems, are introducing new interesting properties derived from execution on dedicated cores. In a previous work [3] [2], we have performed an exploratory analysis of the performance evaluation on virtualized environments extracting preliminary results. In this paper, we deepen into the analysis of DDS in a virtualized deployment, providing a benchmark for the analysis, conducting further experiments, and elaborating conclusions as comparison between the two most popular implementations. We explore the overhead of virtualization in distributed DDS communication stacks by black box benchmarking (with no code fine-tuning), and we reason about the causes of virtualization costs, communication latencies, communication jitter, and execution nondeterminism.

The paper is structured as follows. Section 2 describes related work. Section 3 presents the potential drawbacks of the virtualization technology for timely behavior, and it describes the benchmark elaborated for the experiments or specific tests that have been carried out. In section 4, the proposed virtual data-distribution scenario is defined (two main DDS implementations running on VirtualBox) as well as the used evaluation forms, i.e. processor and network intensive scenarios. Section 5 reports the evaluation results discussing the minimum, maximum, and average response-times in different setups. Finally, section 6 outlines the main conclusions and future work.

II. Related work

Virtualization technology for cloud computing, such as hypervisors and/or virtual machine monitors, can challenge the temporal properties of soft real-time applications due to the possible introduction of higher latencies and communication jitter. Still, the deadlines for the soft real-time domain may be respected (or tolerably lost) by the new high performance cloud computing platforms that provide very efficient networking by using specific technology as InfiniBand [4].

Predictable hypervisors exist that achieve temporal and spatial isolation such as the academic initiatives of [26] [21], among others in the industrial domain[1], for real-time domains. In the hard real-time domain, predictability offered by real-time hypervisors is obtained at the cost of having to recompile the execution environment. This is not desired for the case of soft real-time applications and mainstream domains that are likely to be interested in using existing binaries, and they may even suffer run-time migration.

There are a few studies and analysis of the performance of both, virtualization technology and virtualized environments with varying quality results. Diverse applications have been used as payload to evaluate virtualization performance. These can refer to low-level services [10], function-specific applications (e.g. MapReduce [19] [16], storage solutions [20]), and middleware systems [18]. Some works report [17] significant delays due to the virtualization layer in contexts where applications are in execution within virtual machines. In contrast, other *empty* scenarios (i.e., without applications or virtual machines) report that the execution is similar to the results obtained on the physical platform [24] [25].

For this purpose, other virtualization technologies exist that do not offer temporal isolation but statistical guarantees with the advantage of allowing functional additions at run-time.[2]

The different implementations of DDS were not originally designed for virtualized environments. As a result, they can exhibit a significant different behavior either in a virtualized or in a bare machine with operating system. There are some previous experiences of using DDS in a virtual context offering good average communication times, such as the one reported in the iLAND reference implementation [6] [15] that uses a bi-dimensional QoS model [14] that can be mapped to DDS QoS properties. Possible sources of this behavior are the efficient resource management policies at node level inspired on [12] using QoS resource brokers such as [9]; timeliness was preserved even in the event of system reconfigurations that required real-time service composition [13] [32]. However, no benchmarking was performed in this context and only average times were reported.

Mainstream and traditional individual parallel applications or benchmarks have been applied to evaluating the performance of virtual machines. Benchmarks are being modified to adequately model the operation of virtual machines such as the industry benchmarks VMark [11], vConsolidate [10], and SPEC committee [22] that are virtualization benchmarks that can be used for consistent and repeatable server performance analysis. There are interesting studies applying vConsolidate in specific VM performance modeling such as [27]. Released two weeks prior to the submission of this work, [22] simulates a world-wide company with an IT infrastructure with varied requests that enables specifying deadlines for service requests (from few to hundreds of ms, and supports multiple run configuration for analysing bottlenecks at multiple layers (from hardware to application layer).

The execution of communication middleware in a virtual environment is not supported by a specific benchmark. Consequently, we have identified a set of specific tests for devising the behavior of the system to identify possible bottlenecks, reasoning about the possible sources of the problems.

III. Benchmarking virtualized middleware

The behavior of the system is analysed in terms of *usage of physical resources*, *stability* of the execution, and *load* of the servers is considered as an initial step to analyse the system. Considered resources are: Processor, network bandwidth, and memory consumption. The stability is measured by analysing the behavior of specific communications in the presence of interference and without interference. Different load levels for the servers are also experimented by executing operations that require various resource usage levels, from light weight to

[1] WindRiver Hypervisor, WMWare ESX, etc.

[2] Popular virtualization technologies that provide applications execution environments include Citrix Xen, VMWare, KVM [19], Oracle VirtualBox, SPLPAR, MS Virtual Server and Solaris Container [23].

heavy operations. Other interesting measures are derived such as *throughput* (i.e., number of requests per unit of time), and *latencies*.

A. Potential performance drawbacks in the virtualization

The execution risks of a virtualized communication middleware are the following:

- Overhead of the virtualization. Virtual machines are interfered by the execution of other VMs. This may affect the use of *visible* shared resources (e.g. the same physical core or memory capacity) and *invisible* shared resources (e.g., cache space, memory bandwidth, etc.). These can be visible or invisible depending on the implementation of the host operating system and virtualization monitor.
- Overhead of the communication middleware abstractions. A virtualization infrastructure adds extra costs in the response time of distributed applications since requests traverse the software layers; requests may be queued at different levels. This overhead affects main statistical metrics (i.e. minimum, average, and maximum response times), increasing jitter and overhead. That refers to the cost of serializing and deserializing parameters sent in different communications. Notice that part of this serialization cost may be alleviated using virtual machines that run similar virtualized operating systems and hardware infrastructures.
- Coexistence issues. Other particular inefficiencies stemmed from the integration of two different software stacks: the virtualization software and complex middleware. Depending on the particular middleware-virtualizer combination, different inefficiencies may appear (e.g. unnecessary copies from virtualized buffers to middleware buffers).

B. Benchmark description

In order to produce a meaningful set of tests for virtualized middleware, a benchmark should take into account the following key aspects:

- Application nature. Different types of applications exhibit distinct performance patterns that are, mainly, of two types: (i) *network intensive* applications and (ii) *CPU intensive* applications. Network-intensive applications make heavy use of I/O operations and peripheral actions, and their processor computations are minimum as compared to the network I/O activity. CPU intensive are dedicated to intra-node activity rather than in communication or information exchange.
- Middleware communication paradigm. The supported interaction paradigms of the middleware (e.g. its publish-subscribe (P/S), synchronous remote invocations, etc.) influence its internal implementation and synchronization aspects which directly affects the performance of the remote execution and, as a result, also influence virtualized environments. Other influencing aspects to be taken into account are the marshalling (and unmarshalling) techniques which typically represent a considerable source of overhead in middleware infrastructures.
- Virtualization software characteristics. The type of virtual machine monitor (VMM) or hypervisor and the virtualization technique, and guarantees (either real-time or statistical) over the temporal and spatial isolation of virtual machines influence the performance of the system.

Next section illustrates a practical evaluation via a specific set of tests that consider the above mentioned concepts in a general scenario: i) a client-server application, which is ii) running on DDS, which iii) is virtualized using VirtualBox over Linux. This soft real-time scenario has been chosen because it reduces development and deployment costs (i.e. the time require to develop a virtualized application). Real-time virtualizers would produce better performance results, requiring additional resources (CPU, or additional infrastructure) too.

IV. Analysis of DDS executing in virtual machines

This section describes the set of tests carried out in a client-server application installed on a DDS infrastructure. Such applications are typical of many distributed systems and require the server to block, waiting for a response from the client. In essence, the benchmarked application carries out the following operations:

- The client sends information packed in an array that is transferred to a server node. Internally, the communication with the server is carried out using a DDS topic.
- Then, the server which is another node running DDS, reads the data, processes the data, and sends back a response to the client node. In the specific implementation of the test, this action is supported with a different DDS topic that sends data back to the client.
- After receiving the information, the client to server communications stops so the client-server interaction ends.

A. Experimental setting

The physical deployment comprises two machines, one acting as a server and another as a client (see Table I). Both machines are connected via a local isolated Switched Ethernet network that connects to Linux nodes. Client and server run in a Ubuntu Linux 12.04 virtualized (with Virtualbox) image that communicates via one of two alternative DDS implementations: The first is the OpenSplice 5.5 DDS, and the second is the professional RTI 5.0 implementation.

Since the hosting operating system, the virtualization software and the virtualized operating systems are non real-time infrastructures, the tests carried out focus on average performance that may be suitable in some best-effort real-time applications. A worst-case scenario requires to use a real-time virtualizer and real-time operating system, which are not the focus of this evaluation scenario.

In this particular, the following evaluation goals were pursued:

TABLE I
HARDWARE AND SOFTWARE STACK USED IN THE EVALUATION

HW/SW Item	Description
Server machine: CPU Server machine: Memory	Core2Duo E4500 @2.2 Ghz 6 Gigabyes
Client machine: CPU Client machine: memory	Core2 6320 @1.86 Ghz 3 Gigabyes
Network	100 Mbps switched Ethernet
Hosting OS Virtualization software Hosted OS	Ubuntu 12.04 Virtualbox 4.2 Ubuntu 12.04
First DDS middleware: Second DDS middleware	OSPL Community v5.5.1 RTI Connext Professional 5.0
Small size data sets: Medium size data sets:	64 bytes 512 bytes
Processing time at server:	From 0 to 100 μs

- To measure the absolute performance of client-server applications from different DDS middleware vendors.
- To evaluate the overhead introduced by the virtualization infrastructure in different DDS implementations. To assess the differences in costs introduced by the virtualization process.
- To evaluate the impact of different virtualized DDS middleware implementations from the point of view of a real-time application (considering different deadlines).
- To determine the absolute overhead introduced by the DDS infrastructure when compared against an ideal infrastructure. The ideal infrastructure refers to a minimum distributed system based on ICMP messages that do not pay serialization/deserialization costs.

B. Results and analysis

The first experiment refers to the time required for the whole client-server interaction under different setups. The different setups refer to the following choices:

- The experiment is executing (i) inside the virtual machine or (ii) in the host with no VM intermediation.
- The experiment is running (i) on an ideal ICMP scenario, (ii) on OSPL or on (iii) RTI stacks.
- In the experiment the data sent to the server has to be processed. The processing at the server ranges from 0 to 100 μs.

The obtained results (see Figure 1) show the expected performance patterns. In all cases, the execution costs increase with the amount of data sent to the server. They also increase as they are virtualized, i.e. the costs in the non virtualized environment are less than in the virtualized one, ranging from 800μs to few milliseconds with medium size data sets.

A remarkable result is the gap between the ideal middleware setting (represented in the evaluation with ICMP) and DDS. It is due to the multiple abstractions that are supported by the DDS programming model, mainly due to serialization overhead, and to the use of topics and multiple I/O buffers, that manifest (i.e., are paid for) in the ICMP stack.

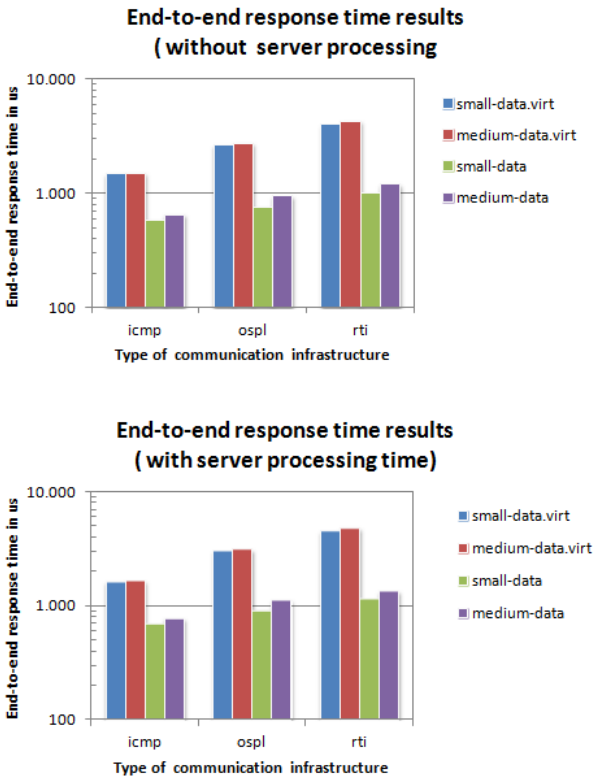

Fig. 1. Absolute end-to-end response time results with server and without server processing time

Figure 2 complements the previous results with information on the extra cost paid by the virtualization process. For the given scenarios, the extra cost ranges from a minimum of 120% to a maximum of almost 300%. In practical terms, the virtualized application has reductions in performance that may leave the available utilization in almost 25% of the time consumed in a non virtualized environment equivalent. Notice that this time is, to some extent, the maximum penalty; this could be alleviated by using optimized virtualizers that take into account the host infrastructure. The virtualizer used in this experiment does not take advantage of this feature to improve performance.

It is also remarkable that the virtualization may require up to 50% of the total available time for small response time applications (i.e., applications with a $10ms$ deadline). This cost is reduced to less than 5% (i.e. a more moderated and admissible penalty) when deadlines are in the $100ms$ range. As operational deadlines increase, this margin reduces to 1% for applications with deadlines that are in the range of milliseconds.

The last set of experiments refers to the overhead introduced by a middleware like DDS. Different middleware implementations introduce an overhead when they compare against an idealized communication middleware that do not require to perform general application serialization, copying data from different multilevel buffer, nor other middleware-

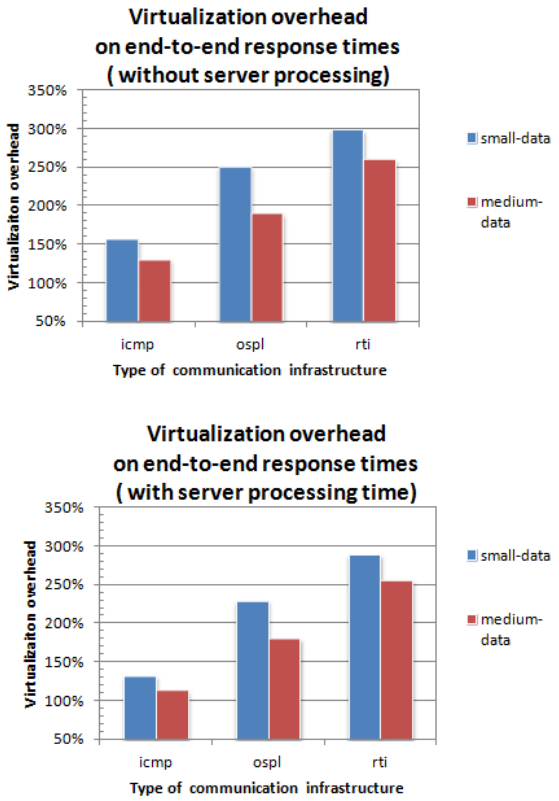

Fig. 2. Virtualization overhead main results with and without server processing time

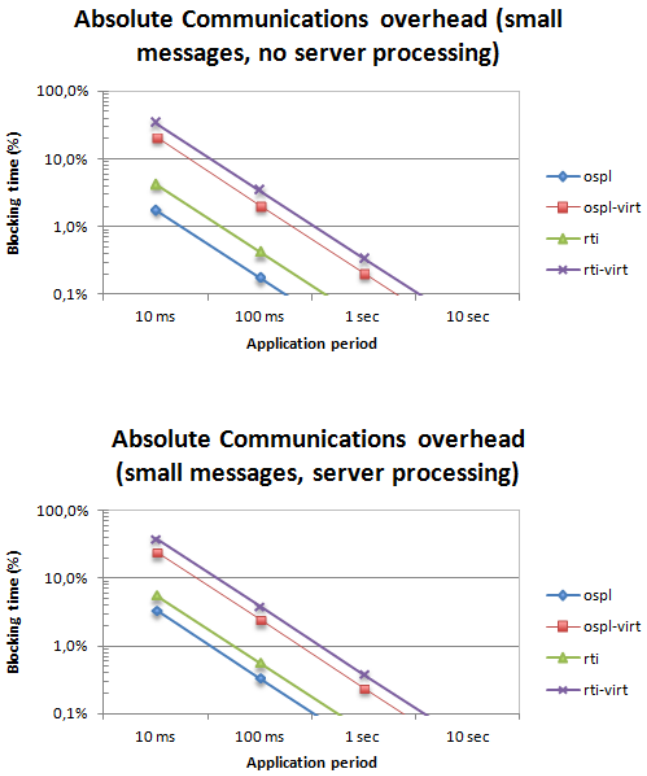

Fig. 3. Overhead introduced by virtualized middleware technology (small size data)

level overhead. As in the previous cases, the evaluation has been carried out in small (see Figure 3) and medium (see Figure 4) data sizes.

The following are remarkable outcomes:

- In most cases, the overhead introduced by the stacks represents an important amount of the available time. This extra overhead takes into account the amount of time required for serialization and deserialization processes.
- For the given virtualization scenarios (and under the described evaluation conditions), the use of OSPL support outperforms an RTI equivalent stack. In average performance terms, the virtualized RTI requires and 50% amount of CPU time to offer an OSPL-equivalent performance.
- Lastly, it should be noticed that for both implementations, the overhead of the virtualization dominates over the overhead introduced by the middleware abstraction. In all tests (see Figure 3 and Figure 4), the cost of the middleware abstraction is typically 30% of the total time, while the the cost of the virtualization may represent 72% of the total time. In practice, this effect is shown in the graphs with the two virtualized DDS-middlewares as virtualized implementations consume more resources than their non virtualized equivalents.

V. Conclusion

The work describes a benchmarking process to obtain information on the performance of virtual machines containing applications that communicate via publish-subscribe (data centric) middleware. Precisely, we have analysed the behavior of DDS for its two most popular implementations (Open Splice and RTI). Initially, we have identified the important aspects to consider in the design of a benchmark for performance analysis of virtualized middleware, including the identification of the potential bottlenecks to search for, and the considerations with respect to the software stack to be analysed. Lastly, we have describe the benchmark tests executed for applications that make intensive use of the network and the processor. Results have shown the comparison and impact on both implementations of the virtualization software.

Future work will include the execution of just released industrial benchmarks for virtual machines that simulate a real environment based on scenarios described in [28], [29] and [30].

Acknowledgment

This work has been partly supported by the Salvador de Madariaga Programme for International Research Stays from the Spanish Ministry of Education (PRX12/00252) and by the Spanish national project REM4VSS (TIN 2011-28339).

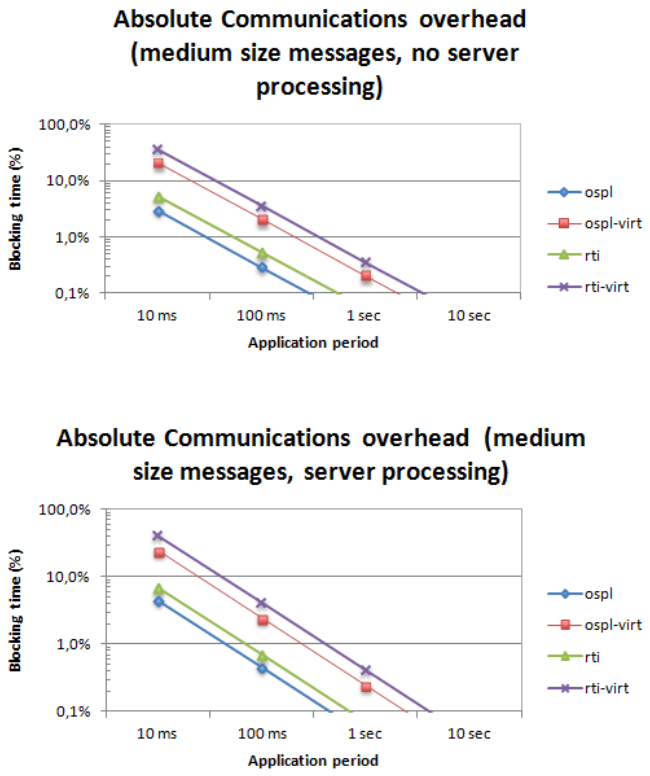

Fig. 4. Overhead introduced by virtualized middleware technology (medium size data)

REFERENCES

[1] Object Management Group. *Data Distribution Service for Real-Time Systems.* OMG, 1.2 formal/07-01-01 edition., Jan. 2007.

[2] R. Serrano-Torres, M. García-Valls, P. Basanta-Val. *Performance Evaluation of Virtualized DDS Middleware* IV Simposio de Sistemas de Tiempo Real (IV STR - CEDI). September 2013.

[3] R. Serrano-Torres, M. García-Valls, P. Basanta-Val. *Virtualizing DDS middleware: performance challenges and measurements*. IEEE International Conference on Industrial Informatics (INDIN'13). Bochum, Germany. July 2013.

[4] Infiniband Trade Association. *InfiniBand Architecture specification vol. 1, release 1.2.1.* 2007.

[5] G. Chen, M. Li, and D. Kotz. *Data-centric middleware for context- aware pervasive computing* Pervasive and Mobile Computing, pp. 216 253, April 2008.

[6] M. García-Valls, L. Fernández Villar, I. Rodríguez López. *iLAND: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time Systems*. IEEE Transactions on Industrial Informatics, vol. 9(1), pp. 228- 236. February 2013.

[7] iLAND project. *iLAND Reference Implementation Installation & User Guide.* September 2012. `http://sourceforge.net/projects/iland-project/`

[8] M. García-Valls, A. Crespo, J. Vila. *Resource Management for Mobile Operating Systems based on the Active Object Model* International Journal on Computer Systems Science and Engineering. July 2013.

[9] M. García-Valls A. Alonso, J. Ruíz, A. Groba. *An architecture for a Quality of Service resource manager for flexible multimedia embedded systems*. In Proc. of 3rd International Workshop on Software Engineering and Middleware (SEM02). Lecture Notes in Computer Science vol. 2596. 2003.

[10] M. Greenfield, J.P. Casazza, and K. Shi. *Redefining server performance characterization for virtualization benchmarking* August 2006.

[11] VMWare. *VMWare vmark*. http://www.wmware.com/products/vmmark/results.html

[12] M. García-Valls, A. Alonso, J. A. de la Puente. *A dual priority assignment mechanism for dynamic QoS resource management*. Future Generation Computer Systems, vol. 28(6), pp.902-911. June 2012.

[13] M. García-Valls, P. Basanta-Val. *A real-time perspective of service composition: key concepts and some contributions*. http://dx.doi.org/10.1016/j.sysarc.2013.06.008 Journal of Systems Architecture, Elsevier. (Available online 16 July 2013)

[14] M. García-Valls, P. Basanta-Val, M. Marcos, E. Estevez. *A bi-dimensional QoS model for SOA and real-time middleware* International Journal of Computer System Science and Engineering, vol. 28. ISSN 0267 6192. September 2013.

[15] M. García-Valls, P. Basanta-Val. *Comparative analysis of two different middleware approaches for reconfiguration of distributed real-time systems*. http://dx.doi.org/10.1016/j.sysarc.2013.08.010 Journal of Systems Architecture, Elsevier. (Available online 24 August 2013)

[16] D. d. Oliveira, K. Ocaña, E. Ogasawara, J. Dias, J. Goncalves, F. Baiao, and M. Mattoso. *Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows* Future Generation Computer Systems, January 2013.

[17] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. G. Shin. *Performance evaluation of virtualization technologies for server consolidation*. HP Laboratories Palo Alto, Enterprise Systems and Software Laboratory, Tech. Rep., 2007.

[18] Oracle. *Middleware virtualization* http://www.oracle.com/us/\-technologies/virtualization/middleware-virtualization-068082.html

[19] A. Matsunaga, M. Tsugawa, and J. Fortes. *Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications*. 4^{th} IEEE International Conference on eScience, 2008. [Online]. Available: http://graal.ens- lyon.fr/ ecaron/m2/papers/papier06.pdf

[20] D. D. Chambliss, G. A. Alvarez, P. Pandey, D. Jadav, J. Xu, R. Menon, and T. P. Lee. *Performance virtualization for large-scale storage systems*. 22^{nd} Int'l Symposium on Reliable Distributed Systems. 2003.

[21] S. Xi, J. Wilson, C. Lu, and C. Gill. *RT-Xen: Towards real-time hypervisor scheduling in Xen*. Washington University in St. Louis, Tech. Rep. http://www.cse.wustl.edu/lu/papers/emsoft11.pdf. 2011.

[22] Standard Performance Evaluation Corporation. *Specjbb2013* http://www.spec.org/jbb2013/ 2013.

[23] E. Benjamin. *3 key trends in middleware virtualization*. VMWare, vFabric Team, Tech. Rep. http://blogs.vmware.com/vfabric\-/2012/08/3-key-trends-in-middleware-virtualization.html 2012.

[24] J. Lu, L. Makhlis, and J. Chen. *Measuring and modeling the performance of the Xen VMM*. bMC Software Inc.

[25] J. Lei, X. Yang, G.Xiong, W. Jiang, and Y.Liao. *VMM-based real-time embedded system* International Conference on Embedded Software and Systems Symposia, pp. 213218. July 2008.

[26] A. Crespo, I. Ripoll, and M. Masmano. *Partitioned embedded architecture based on hypervisor: The XtratuM approach*. European Dependable Computing Conference, pp. 67-72. 2010.

[27] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell. *Modeling virtual machine performance: challenges and approaches.* SIGMETRICS Perform. Eval. Rev. vol.37(3), pp. 55-60. January 2010.

[28] P. Basanta Val, M. Garcia-Valls, M. Baza-Cuado. *A simple data-muling procotol*. Accepted in, IEEE Transactions on Industrial Informatics. 2013.

[29] P. Basanta Val, M. Garcia-Valls. *A Distributed Real-Time Java-centric Architecture for Industrial Systems*. Accepted in, IEEE Transactions on Industrial Informatics. DOI: 10.1109/TII.2013.2246172. 2013.

[30] P. Basanta Val, M. Garcia-Valls. *Resource Management Policies for Real-time Java Remote Invocations*. Accepted in, Journal of Parallel and Distributed Computing. DOI: 10.1016/j.jpdc.2013.08.001. 2013.

[31] P. Basanta-Val, M. Garca-Valls. *Enhanced JRMP multiplexing headers under non-fragmented local area network constraints*. Electronics Letters, vol.49, no.21, pp.1333-1335. October 2013. doi:10.1049/el.2013.2239

[32] M. García-Valls, P. Basanta-Val. *Real-time reconfiguration in Complex Data-Oriented Applications*. Future Generation Computer Systems, Elsevier. (Accepted for publication: October 2013)